ORB ACADEMIC PUBLISHER
Int. J. Comput. Sci. Theor. App., 2015, vol. 4, no. 2., p. 40-46.
Available online at www.orb-academic.org
International Journal of Computer Science: Theory and Application
ISSN: 2336-0984

Solution For Cloud Databases' Privacy Issue

Artem Sumaneev* and Kirill Shatilov

Department of Information Technology, Novosibirsk State University, Novosibirsk, Russia
Email: sumaneevartem@gmail.com, shatilov@ccfit.nsu.ru
** Corresponding author.*

ABSTRACT
Cloud computing and, in particular, cloud databases are useful tools providing flexibility for enterprise's IT infrastructure. But usage of such technology may lead to privacy issues because users have to entrust to cloud services. In this paper we propose a solution for secured cloud database without significant limitations. Our approach is based on usage of following functional elements: SQL-queries and response processors, cryptographic metadata storage and variety of encryption algorithms. Usage of different encryptions like order-preserving and homomorphic allows performing resource-intensive computations on cloud server on encrypted data. SQL-queries processor transforms user's queries into a secured form, analyzing query and encrypting vulnerable data. Response processor decrypts a secured data in database's response and converts it to a suitable for user form. Cryptographic metadata storage provides access to encryption keys. Architecture with detailed description of components and theirs interaction's principles will be presented. Our approach has been validated by an implementation of a prototype and its integration with WordPress Content Management System.

KEYWORDS
DbaaS — Privacy in Cloud — Order Preserving/Homomorphic Encryption — SQL.

1. Introduction

Rapid growth and development of cloud technologies has leaded them to popularity and wide spread usage. Although customers are excited by cloud features and benefits, they are very concerned about confidentiality of data stored and processed in a cloud. Insider threats combined with a general lack of transparency into provider process and procedure has dropped confidence in security of cloud data storage [1].

Data confidentiality is highly important for Cloud Databases (Database as a Service, DbaaS), and there are threats of disclosure of vulnerable user's data to unauthorized parties. First of all, curious and malicious database administrators may capture or leak data [3]. Also a theft of database may possibly occur, leaving data in hands of a malefactor [4].

Listed problems are still actual [2], and as a result multiple solutions to problem of trusting clouds have been developed. Encryption of all data in remote database was offered as a method of providing provable confidentiality [5, 6]. But such an approach demands all operations will be held on a client side after decryption of database content. Other solutions such as MIT CryptDB [7], lack fully homomorphic encryptions and use third-party encryptions with relatively low crypto strength and known vulnerabilities [8].

To address listed cloud security issues we designed secure cloud database architecture, several encryption algorithms and a SQL data encoding component. Proposed solution addresses mentioned challenges using following key ideas:

1. Incoming queries are processed in secured form by a trusted proxy server. No encryption keys are passed to untrusted database or leaves trusted area in other way.
2. Usage of wide variety of order preserving and fully homomorphic encryptions. All encryptions that are used in proposed solution are proprietary developed encryptions with relatively high speed and provable crypto strength.
3. All resource-intensive operations except encryption and decryption are performed in a cloud on encrypted data without decryption step due to used encryption algorithms.

In this paper we present upgraded and extended version of solution described in [10], its work's principles, design, SQL queries' processing and security of cryptographic metadata storage.

The next section of paper gives an introduction into principles solution's functioning cycle and depicts main idea of SQL queries handling. Next, Section 3 describes the architecture of proposed solution, also summary for each component is given. SQL-queries processor and cryptographic metadata storage's

principles are explained in the Sections 4 and 5 correspondingly. Section 6 gives some insight into encryptions, which are used in secure SQL queries processing. Finally, the last section summarizes all achievements of proposed secured cloud database concept.

2. Basic Principles

In this Section basic principles are discussed illustrating main idea of secure cloud database. Core component of proposed secure cloud database is SQL-queries processor. All user's SQL queries are analyzed and transformed by program components in trusted area before sending to DBMS (see Figure 1). Similarly, all replies from database are processed.

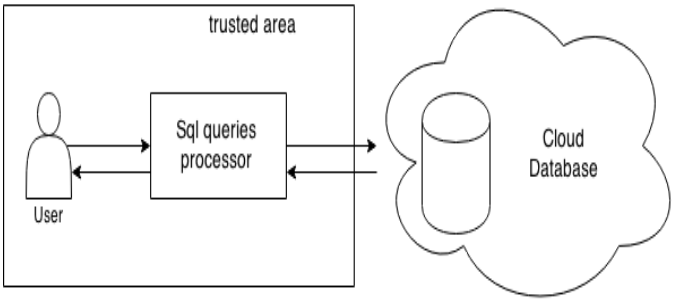

Figure 1. *Overview.*

Also, encryption of data, extracted from user's queries, is a responsibility of SQL-query processing component. Proposed solution doesn't require any modifications on DBMS side.

User's SQL query handling is shown on Figure 2. Information about encrypted columns such as encryption keys and encrypted column name(s) is needed in order to parse, decrypt and reconstruct user queries.

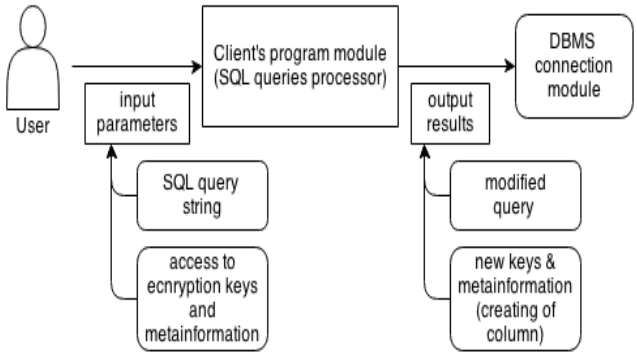

Figure 2. *Query processing.*

When user retrieves data from a cloud database, selection of column comes as a reply from DBMS. Information about encrypted columns is needed to decrypt and present selected data to user in suitable form (see Figure 3).

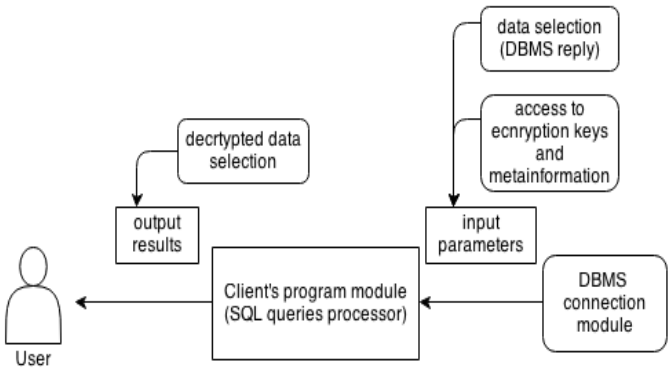

Figure 3. *Data selection processing.*

Basic idea of SQL-queries processing is explained on different types of SQL expressions below. "CREATE" statement is the only statement in terms of proposed secure cloud database, which may contain additional keywords, which are not included in SQL language. These keywords are markers of different encryptions applied to table columns. To indicate that column in currently created table should be encrypted, user should add a constraint corresponding to encryption's marker (identification string).

If SQL-queries processing component encounters encryption marker, following steps will be performed. First, encryption's keys are generated or chosen. Next, based on encryption information, number (can be more than one, in case when the result of encryption is a vector of multiple values), names, types and constraints of output columns are determined. Correct SQL string is created according to determined information. After that modified statement is sent to DBMS. Output names of encrypted columns are anonymized, while anonymization of table name is optional (anonymization means changing real names of column to generated ones).

Processing of SQL statements such as "INSERT", "SELECT", "DELETE" uses common principles. All data from query are extracted and data from encrypted columns are encrypted. Also names of columns, which values are encrypted and used in processed statement, are modified according to their anonymized names. In some cases (for example, homomorphic encryption) changes in mathematical operation can be made. Responses for "SELECT" queries from database are decrypted if needed.

Correctness of performing "JOIN" operation inside DBMS depends on encryption properties. If encryption is deterministic, output column is single and both columns from each joining tables, have same key, no additional mechanisms are needed to perform "JOIN" operation. It is very important to understand that some restrictions may apply to using full functionality of SQL language and different DBMS specific structures due to fact that proposed solution targets multi DBMS support, also various constraints can be caused by using order preserving or

fully homomorphic encryption.

One of the restrictions is limited usage of "ALTER TABLE" construction. As long as table altering doesn't affect encrypted columns, it can be performed, but adding or removing encryption from already existing table is unsupported. Another restriction is incompatibility of encryptions with several column constraints (e.g. "FOREIGN KEY").

This concludes basics principles and mechanisms of SQL-queries processing in discussed approach. Main idea is to perform query analysis and modification, which include encryption of vulnerable user's data on client side, without affecting DBMS or adding any intermediate components.

3. Architecture

This section gives an insight into each component of the presented solution. Cloud DBMS remains unmodified; all other components are located in trusted area, as it shown in Figure 4.

Basic principle is to intercept queries for secured database and its responses, modify and send them to the destination. For this proxy server is used as an intermediate component between user and DBMS. This component is responsible for following functions:

1. Intercepting user's queries, modifying them via SQL processor and sending encrypted queries to secured database.

2. Intercepting DBMS's responses, decrypting them using Response processor and forwarding decrypted responses to users.

3. Handling of user's authorization in cryptographic metadata storage.

Second part is SQL-queries processor. Its main purpose is to convert user's query using cryptographic metadata storage and encryption algorithms and so it can be performed on secured data in database. Basic principle of processing is to find columns used in query and check if metadata storage contains information about them or that there are keywords specifying encryptions in processed query. After that SQL processor changes this column on their anonymous names, encrypts values used with column in operations and transforms operations into secured form. Mechanism of SQL queries' processing is better described in Section 4.

Next observed basic component is Database response's processor. Its main purpose is to detect encrypted columns in response, combine them (when multiple output columns correspond to single initial column), and to decrypt data to display them in suitable form to user. This module actively interacts with Cryptographic metadata storage, in order to correctly decrypt and modify response.

Cryptographic metadata storage is responsible for storing information supporting SQL queries and DBMS replies processors. Among service information, the following values are kept in this storage:

- crypto keys for encryption of data in column
- map of real name of the column to anonymized names matching vector of encrypted values
- types of encryption used for column
- names of tables, where encrypted columns are located

Cryptographic metadata storage is an interface for retrieving and adding information about encrypted columns. It's mechanism is shown in Section 5.

Encryptions interfaces module provides two interfaces — "Key" and "Encoder". These interfaces define a set of properties required for encryptions' correct work and interaction with other components. Due to "Encoder" and "Key" interfaces architecture and realization of entire solution does not depends on specific encryptions and is open to integration with other crypto algorithms.

Arithmetic operations on encrypted values may be represented with some specific form depended on algorithm. That is why algorithm must be considered in transforming of operations with secured data.

Each encryption is able to perform fixed subset of operations (plus, multiplication, comparing etc...). Hence operation with secured data's conversion in SQL-queries processor must consider for operation and encryption's compatibility.

4. SQL-queries Processor

Current section introduces in SQL-queries processor's mechanism shown in Figure 5.

SQL-queries processor's main objective is modification of incoming queries, which consists of following steps:

1. First step is to parse not-secured query and build abstract syntax tree (AST) of query where each node corresponds to an operation.

2. After parsing Processor searches for secured columns which have records in cryptographic metadata storage. Simultaneously, Processor checks for correct usage of secured data within construction. For example there is no way to use columns with different specified encryptions in one arithmetic operation. Additionally, it must be checked if operation cannot be implemented on secured data depending on encryption. As an example a plus-operation cannot be performed if encryption algorithm is not additively homomorphic.

3. When all secured columns and used with them values are found, they are changed on generated anonymous columns' names and encrypted value, accordingly. Operations are converted to secured form. Transformed operation from AST's root corresponds to secured query.

While "CREATE TABLE" query is processed newly created secured columns marked with keyword specifying encryption

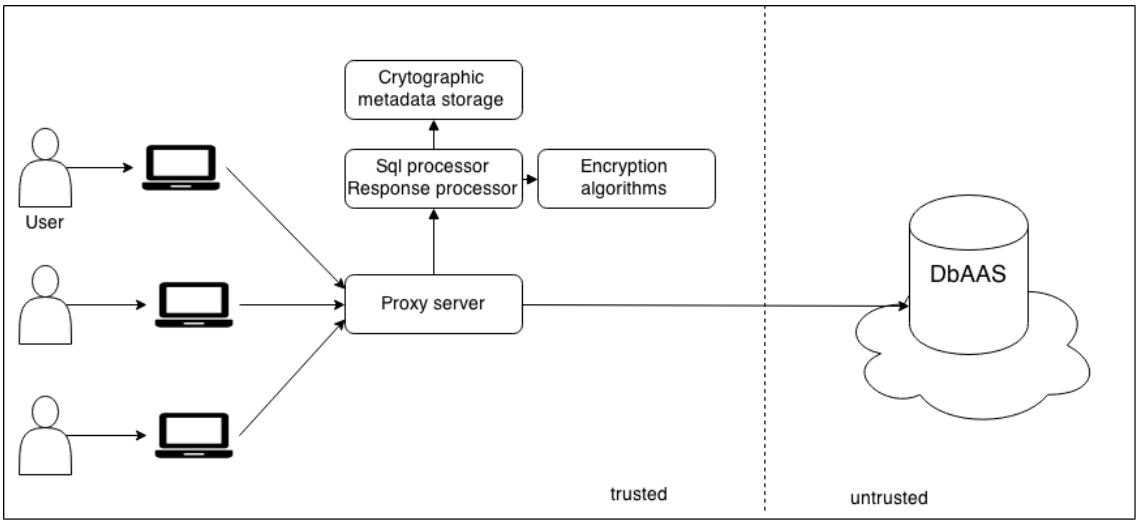

Figure 4. *Solution's architecture..*

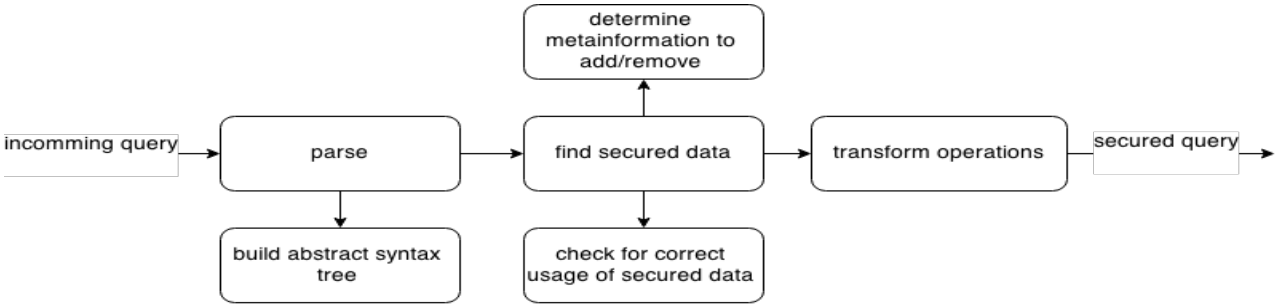

Figure 5. *SQL-queries processor's functioning.*

may be found. This secured columns demands metadata for correct usage of them. Therefore, SQL-query processor retrieves data from query which is column name, column's table name, column's type and encryption algorithm specified by marker to create new record in cryptographic metadata storage.

On the contrary, "DROP TABLE" query removes table from database. So all records in storage associated with dropped table will be erased.

For example, user issues the following query:

CREATE TABLE tbl (id INT),
(price INT enc_ord NOT NULL)
(count INT encr_homomorphic)
CHECK (price≥0)

Column definition of id column will not be modified because there is no secure marker specified. In second definition processor will find keyword for order-preserving encryption and add new record in cryptographic metadata storage for price column where names and types of encrypted column will be generated by encryption type. Processed definition looks like

(encr_price_0 INT NOT NULL),
(encr_price_1 INT NOT NULL),...

where *encr_price_x* is a newly generated name. The count column definition will be processed in the same way, only the record for this column will contain information about homomorphic encryption.

Next, CHECK-constraint will be modified. Processor will use newly created record to transform CHECK-expression into the secured form with check if encryption algorithm for price supports greater-than operation:

CHECK ((encr_price_0≥encrypt_ord(0)[0])
AND (encr_price_1≥ encrypt_ord(0)[1]) AND...)

where *encrypt_ord(0)* is encrypted values' vector of 0. Final query will be:

CREATE TABLE tbl (id INT),
(encr_price_0 INT NOT NULL),
(encr_price_1 INT NOT NULL),...(encr_count_0 BLOB),
(encr_count_1 BLOB),...
CHECK ((encr_price_0≥encrypt_ord(0)[0])
AND (encr_price_1≥encrypt_ord(0)[1]) AND...)

Let us consider a SELECT query received after table creation:

SELECT SUM(count) FROM tbl

Firstly, after parsing, the processor will check if the column *count* from the table *tbl* has its record in cryptographic metadata storage and its encryption supports additive operation. A record was created in previous example and encryption is additive homomorphic for demonstration. Also is considered, that SUM operation is simple column-wise sum. Query will be transformed:

SELECT
SUM(encr_count_0),SUM (encr_count_1),...
FROM tbl

The Response processor will detect in DBMS's response names of encrypted columns, find appropriate decryptor storage's record corresponding to that names and decrypts response's values from encrypted columns. Processed response will be sent to user.

5. Cryptographic metadata storage

Section 5 presents functioning cycle of cryptographic metadata storage and its security. Cryptographic metadata is necessary for correct modifying of user's queries as it was shown in previous sections. Each record for secured columns contains such information as table and column's names, encryption type, encryption keys etc. Because of storing encryption keys storage must be protected.

To ensure security of the storage the following technique was used. While application is running all metadata are located in RAM. When a new record is added to the storage the backup of whole storage is created encrypted with one of encryption algorithms. Upon application's shutdown RAM is cleared and there is last valid encrypted backup in NVRAM. This prevents an unauthorized access, insider's malicious actions and power failure of proxy server.

In addition another advantage of such technique is performance of encryption key's extracting which boost encryption and decryption activities.

6. Encryptions

This section features description of encryptions that secure cloud database uses. We use three types of encryption: deterministic, order preserving and homomorphic encryptions.

Deterministic. Deterministic encryption provides strong security, it leaks only which encrypted values correspond to the same data value. In secure cloud database it can be used for storing password hashes, when no operations are conduct over data, but confidentiality is very important.

In proposed solution, we use proprietary developed deterministic block encryption [11].

Order Preserving. Order preserving (OP) encryption allows order relations between encrypted data items to be established, without revealing data itself. Such encryption can be used to protect salaries or other economical information inside secure database with possibility of performing order operations.

There are various OP encryptions, used in solution. For different types of data we can use different OP encryptions in single table; this provides extra resistance to crypto attacks. Only two encryption schemes will be discussed in this paper.

6.1 Radix encryption

The second scheme's, *based on different number systems*, basic idea is conversion of numbers from notation with one radix to another.

For the first step, it is necessary to obtain the vector of coefficients from number in first-radix representation. Next step is replacing in the current representation first radix with second chosen radix. At the last step performed when a subsidiary vector of nonnegative numbers is added to the vector of coefficients from the current number representation. Note that values of the sum of these vectors must be less than second radix and second radix is greater than first.

Having final representation with new radix and modified coefficients the result can be calculated as second-radix – decimal conversion. The secret key is consist of first, second radixes and subsidiary vector of nonnegative numbers.

To illustrate this idea, let us consider one iteration of encryption. There can be made several iterations. Original number is $S \in N$.
Secret key is:

$$p, q \in N, \langle b_0, b_1, ..., b_{n-1} \rangle, b_i < q, b_i \in N$$

Steps of encryption:

1. Obtaining S's p-radix representation:

$$s = \alpha_0 + \alpha_1 * p + \alpha_2 * p^2 + ... + \alpha_{n-1} * p^{n-1}$$

2. Replacing p-radix with q-radix:

$$s' = \alpha_0 + \alpha_1 * q + \alpha_2 * q^2 + ... + \alpha_{n-1} * q^{n-1}$$

3. Adding subsidiary vector $\langle b_0, b_1, ..., b_{n-1} \rangle$ to the vector of coefficients $\langle \alpha_0, \alpha_1, ..., \alpha_{n-1} \rangle$

$$s" = (\alpha_0 + b_0) + (\alpha_1 + b_1) * p + (\alpha_2 + b_2) * p^2 + ...$$
$$+ (\alpha_{n-1} + b_{n-1}) * p^{n-1}$$

where $\forall i : \alpha_i + b_i < q$.

The result of encryption is $\omega = s''$.
Process of decryption consists of following steps:

1. Obtaining ω's q-radix representation:

$$\omega = y_0 + y_1 * q + y_2 * q^2 + ... + y_{n-1} * q^{n-1}$$

2. Replacing q-radix with p-radix:

$$\omega' = y_0 + y_1 * p + y_2 * p^2 + ... + y_{n-1} * p^{n-1}$$

3. Subtracting vector $\langle b_0, b_1, ..., b_{n-1} \rangle$ from the vector of coefficients $\langle y_0, y_1, ..., y_{n-1} \rangle$:

$$\begin{aligned} \omega\text{"} &= (y_0 - b_0) + (y_1 - b_1) * p + (y_2 - b_2) * p^2 + ... \\ &+ (y_{n-1} - b_{n-1}) * q^{n-1} \end{aligned}$$

The result of decryption is , which is equal to S.

Let encrypt a number $N = 115$ with key $(2,3,1,0,2,1,0,1,1)$ (one iteration). First step is to transform N to p-radix representation where $p = 2$:

$$S = 115 = 1*2^0 + 1*2^1 + 0*2^2 + 0*2^3 + 1*2^4 + 1*2^5 + 1*2^6$$

Then vector of coefficients is $= (1,1,1,0,0,1,1)$. New radix is $q = 3$ and subsidiary vector is $B = (1,0,2,1,0,1,1)$ because of chosen encryption key. Second and third steps are to add coefficients' vector to subsidiary vector.

$$\begin{aligned} S\text{"} &= (1+1)*3^0 + (1+0)*3^1 + (0+2)*3^2 + (0+1)*3^3 \\ &+ (1+0)*3^4 + (1+1)*3^5 + (1+1)*3^6 \\ &= 2075 \end{aligned}$$

The number S" is a result of encryption.
Now let us process decryption of found number W. Firstly we convert this number to numeral system with radix $q = 3$:

$$\begin{aligned} W &= 2075 \\ &= 2*3^0 + 1*3^1 + 2*3^2 + 1*3^3 + 1*3^4 + 2*3^5 + 2*3^6 \end{aligned}$$

Next we subtract coefficients' vector $(1,0,2,1,0,1,1)$ from subsidiary vector $(2,1,2,1,1,2,2)$:

$$\begin{aligned} W' &= (2-1)*3^0 + (1-0)*3^1 + (2-0)*3^2 + (1-1)*3^3 \\ &+ (1-0)*3^4 + (2-1)*3^5 + (2-1)*3^6 \\ &= 1*3^0 + 1*3^1 + 0*3^3 + 1*3^4 + 1*3^5 + 1*3^6 \end{aligned}$$

After that we change radix to $p = 2$:

$$\begin{aligned} W\text{"} &= 1 + 2^0 + 1 + 2^1 + 0 + 2^3 + 1 + 2^4 + 1 + 2^5 + 1 + 2^6 \\ &= 115 \end{aligned}$$

which is origin number N.
The algorithm of encryption is correct and order preserving. Modification of the considered scheme was used in the implementation. There are several iterations; also number of bits for values in the key can be specified in encryption module configuration.

This encryption has passed multiple tests and following results were measured:
Speed of encryption $125 Mbit/s$
Speed of decryption $111 Mbit/s$
(PC's configuration: Mobile Dual Core Intel Atom N570, 1666 MHz, 4 GB RAM, OS Windows 7).
Homomorphic. An encryption scheme is called fully homomorphic if it's able to evaluate an arbitrary function over ciphertexts. In this case decrypted value must match to a calculation result of the same function over plaintexts. The main feature of scheme [12] that is used in proposed secure cloud database is ability to define a strict upper bound of ciphertext size when performing calculations on it for both addition and multiplication.

7. Conclusion

In this paper solution for ensuring data privacy on an untrusted database server is presented. Proposed solution achieves this goal using following key ideas:

- All confidential data, stored in untrusted database, are encrypted. Neither not-encrypted confidential data nor encryption keys leave trusted area.
- Modified queries are efficiently executed in DBMS on encrypted data due to usage of order-preserving and homomorphic encryption algorithms. Trusted proxy server carries out encryption and decryption operations.
- SQL-processor and response-processor are used to transform incoming queries and database's responses. Cryptographic metadata storage handles and stores data about secured columns.

To ensure prototype of solution works properly, it was integrated into WordPress Content Management System. Integration consisted in redirecting connection with MySQL to prototype as proxy between WordPress and database. Also markers specifying encryption algorithms were added to initial data scheme. Tests on queries generated by WordPress show that processing of queries and DBMS's responses is correct.

Future development of solution lies in following items:

- Integration with WordPress shows that alias construction may be required for secured columns in existed program products. Hence this operation must be implemented in SQL and response processors.
- Development and improvement existing encryptions. Speed optimization is one of the most significant goals.
- Security improvement. Interest in analysis of encryption weaknesses and vulnerabilities [8, 9] is escalating, thus several measures can be taken to minimize risk of successful security breach. For example, to complicate frequency analysis, subsystem of phantom "SELECT" queries can be made in order to average number of queries to each column. Another idea for improving system's resistance to attacks is to add to columns garbage data that will be detected and ignored during decryption on client side. This method can change distribution of encrypted data massive inside DBMS, and as a result can make more difficult crypto attack on OP encrypted columns.

References

[1] Cloud Security Alliance. Top Threats to Cloud Computing V1.0 Cloud Security Alliance 2010.

[2] Cloud Security Alliance. The Notorious Nine. Cloud Computing Top Threats in 2013. Available from: `https://downloads.cloudsecurityalliance.`

org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf

[3] William R Claycomb, Alex Nicoll: Insider Threats to Cloud Computing: Directions for New Research Challenges CERT 2012.

[4] Privacy Rights Clearinghouse. Chronology of data breaches. Available from: http://www.privacyrights.org/data-breach

[5] FELDMAN, Ariel J., ZELLER, William P., FREEDMAN, Michael J., et al. SPORC: Group Collaboration using Untrusted Cloud Resources. In : *Proceedings of the 9th Symposium on Operating Systems Design and Implementation*. 2010. p. 337-350.

[6] MAHAJAN, Prince, SETTY, Srinath, LEE, Sangmin, et al. Depot: Cloud storage with minimal trust. In: *Proceedings of the 9th Symposium on Operating Systems Design and Implementation*, Vancouver, Canada, October 2010.

[7] POPA, Raluca Ada, REDFIELD, Catherine, ZELDOVICH, Nickolai, et al. CryptDB: protecting confidentiality with encrypted query processing. In : *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011. p. 85-100.

[8] XIAO, Liangliang, YEN, I., et al. Security analysis for order preserving encryption schemes. In : *Information Sciences and Systems (CISS), 2012 46th Annual Conference on*. IEEE, 2012. p. 1-6.

[9] STEINWANDT, Rainer, GEISELMANN, Willi, et END-SULEIT, Regine. Attacking a polynomial-based cryptosystem: Polly Cracker. *International Journal of Information Security*, 2002, vol. 1, no 3, p. 143-148.

[10] SHATILOV, Kirill, BOIKO, Vladislav, KRENDELEV, Sergey, et al. Solution for secure private data storage in a cloud. In : *Computer Science and Information Systems (FedCSIS)*, 2014 Federated Conference on. IEEE, 2014. p. 885-889.

[11] EGOROVA, V., CHECHULINA, D., KRENDELEV, S. F. New View on Block Encryption (Unpublished), 2003. Available: https://db.tt/vnE9wfgj

[12] ZHIROV, Alexander, ZHIROVA, Olga, et KRENDELEV, Sergey F. Practical fully homomorphic encryption over polynomial quotient rings. In : I*nternet Security (WorldCIS), 2013 World Congress on*. IEEE, 2013. p. 70-75.